LiDAR
Light Detection And Ranging

WeGo Korea

ROS

WeGo

목차

ROS

WeGo

03

LiDAR make_pkg

ROS

WeGo

1. ROS

catkin Package 어떻게 구성이 되어있나?

- 패키지는 반드시 몇 가지 조합에 부합
- 패키지는 반드시 catkin compliant package.xml 파일을 포함

→ package.xml 파일은 패키지의 메타 정보를 제공

(패키지 이름, 버전 번호, 작성자, 관리자 및 파일 종속성)

- 패키지는 반드시 catki에서 쓰이는 CmakeList.txt 파일을 포함

→ 없는 경우: 패키지를 사용하지 않겠다.

1. ROS

Catkin

- ROS가 설치되면 Catkin이 기본적으로 포함

Catkin_package

1. catkin의 작업공간이 ex) catkin_ws에 있다고 가정하면
   작업공간의 루트에서 catkin_make를 호출

1. ROS

Creating a catkin Package

1. Creating a Workspace

```
$ cd ~/catkin_ws/src
```

2. Catkin_create_pkg [catkin_creat_pkg <pakge_name> [depend1] …

```
$ catkin_create_pkg ros_start std_msgs rospy
```

3. Building a catkin workspace and sourcing

```
$ cd ~/catkin_ws
$ catkin_make                (build)
```

ROS

WeGo

1. ROS

Writing Publisher and Subscriber

1. Code

$ mkdir scrpts

$ cd scripte

1. ROS

Writing Publisher and Subscriber

1. Publisher

```
#include “ros/ros.h”
//[ROS SYSTEM necessary to use the most common public]
#include “std_msgs/String.h”
#include <sstream>
int main(int argc, char **argv)
  ros::init(argc, argv, “talker”);
  ros::NodeHandle n;
  ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);
  ros::Rate loop_rate(10);
  int count = 0;
```

ROS

WeGo

1. ROS

Writing Publisher and Subscriber

1. Publisher

```
int count = 0;
while (ros::ok())
{
    std_msgs::String msg;
    ss << "hello world " << count;
    msg.data = ss.str();
    ROS_INFO("%s", msg.data.c_str());
```

1. ROS

Writing Publisher and Subscriber

1. Publisher

```
        ROS_INFO("%s", msg.data.c_str());
        chatter_pub.publish(msg);
        ros::spinOnce();
        loop_rate.sleep();
        ++count;
    }
    return 0;
}
```

1. ROS

Writing Publisher and Subscriber

1. Subscriber

```
#include "ros/ros.h"
#include "std_msgs/String.h"

void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
  ROS_INFO("I heard: [%s]", msg->data.c_str());
}
```

1. ROS

Writing Publisher and Subscriber

1. Subscriber

```
int main(int argc, char **argv)
{
    ros::init(argc, argv, "listener");
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);
    ros::spin();
    return 0;
}
```

1. ROS_Python

Writing Publisher and Subscriber

1. Publisher

```
#!/usr/bin/env python
# license removed for brevity
import rospy
form std_msgs.msg import String
```

1. ROS_Python

Writing Publisher and Subscriber

1. Publisher

```
def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker' , anonymous=True)
    rate = rospy.Rate(10)
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        rate.sleep()
```

1. ROS_Python

Writing Publisher and Subscriber

1. Publisher

```
if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterrupExceptions:
        pass
```

1. ROS_Python

Writing Publisher and Subscriber

1. Subscriber

```
#!/usr/bin/env python
import rospy
form std_msgs.msg import String
```

1. ROS_Python

Writing Publisher and Subscriber

1. Subscriber

```
def listener():
        rospy.init_node('listener', anonymous=True)
        rospy.Subscriber("chatter", String, callback)
        rospy.spin()
if __name__ == '__main__':
        linstener()
```

1. ROS

Publisher and Subscriber (C++)

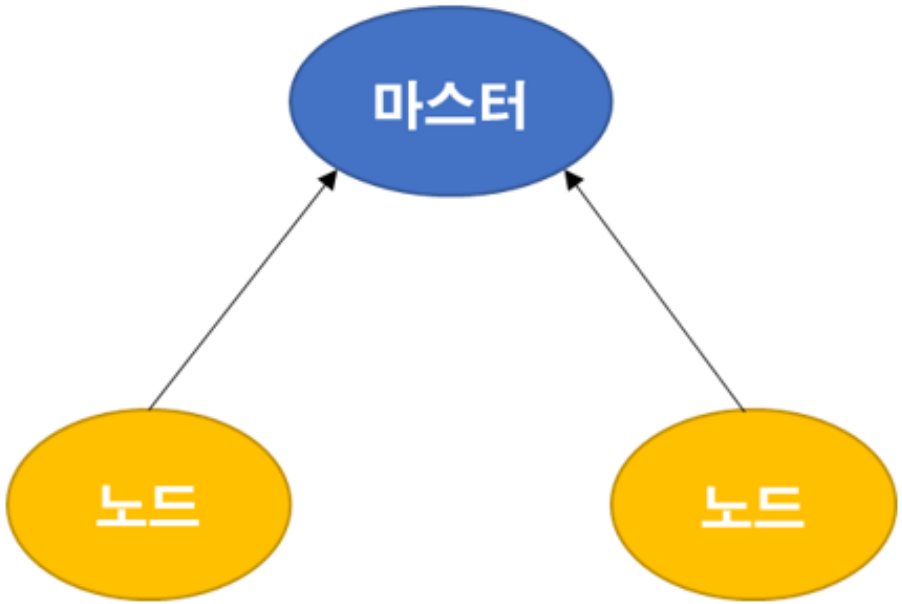
1. rosrun [pkg_name] [depend] - Publisher
2. rosrun [pkg_name] [depend] - Subscriber

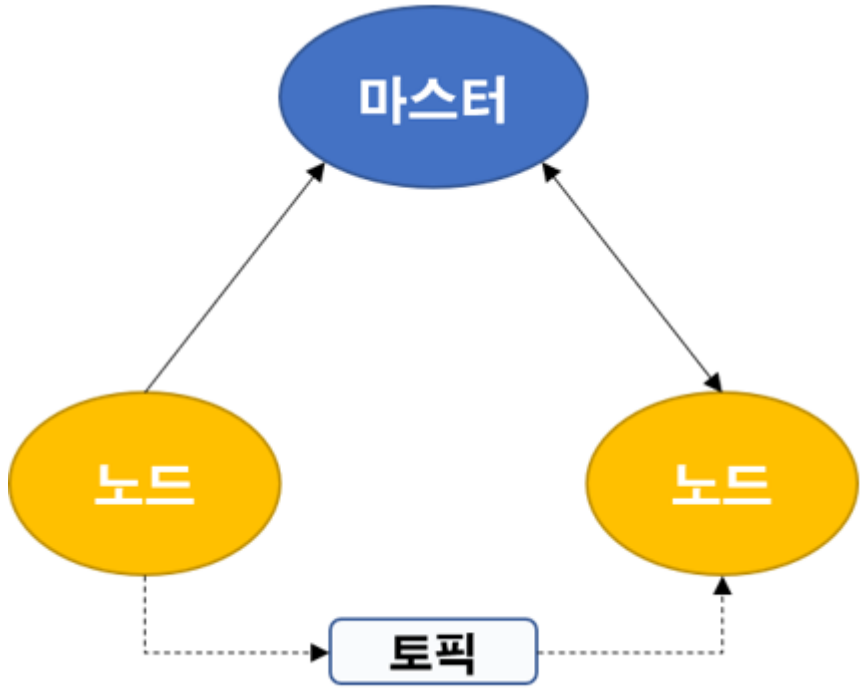
Publisher and Subscriber (Python)

1. rosrun [pkg_name] [depend.py] - Publisher
2. rosrun [pkg_name] [depend.py] - Subscriber

ROS

WeGo





ROS

WeGo

Thanks for your attention!

ROS

WeGo